REMARKS

The comments of the Applicant below are each preceded by related comments of the Examiner (in small, bold type).

> **Claims 26-29, 45 and 59 are rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the written description requirement. The claim(s) contains subject matter which was not described in the specification in such a way as to reasonably convey to one skilled in the relevant art that the inventor(s), at the time the application was filed, had possession of the claimed invention.**
>
> **Claim 26 recites "[a] machine-accessible medium, which when accessed results in a machine performing operations". The specification fails to disclose any medium which can be construed as a machine-accessible medium which when accessed results in a machine performing operations.**

Claims 26-29 as originally filed recite a computer program product residing on a computer readable medium for causing a parallel processor to perform certain functions. In this Reply, the "Description" section has been amended to include a paragraph describing the computer program product recited in the original claims 26-29. Since the originally filed claims are part of the specification, no new matter has been added. Applicant submits that the amended specification fully supports claims 26-29 that are currently pending.

> **Claims 28-29 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.**
>
> **Claims 28 and 29 recite the limitation "[t]he computer program product of claim 26" on line 1 of each claim. There is insufficient antecedent basis for this limitation in the claim. Appropriate correction is required.**

Claims 28 and 29 have been amended.

> **Claims 13, 49, 55, 56 and 61-65 are rejected under 35 U.S.C. 102(e) as being anticipated by Correale, Jr. et al. [US 6,587,905 Bl] (hereinafter "Correale").**
>
> **Per claims 13, 49, 55, 56 and 61, Correale teaches a data processor (see Correale, claim 4, "a data processing system", the processor is considered to be including the CPU, other Master devices, and the PLB Arbiter) comprising:**
>
> **a plurality of programming engines (see Fig. 10, the Master devices);**
>
> **a push arbiter (see Fig. 10, the PLB Arbiter) to arbitrate use of an unidirectional push bus (see Fig. 10, rdDBus and rdDBusAux) by a plurality of external memory resources that are external to the data processor (see Fig. 10, the slave devices) in which requests for using the push bus are sent from the memory resources (when an unbalance in read and write traffic occurs, an Auxiliary_read 104 signal is broadcast to all slave devices, and one of the slave devices claims this read cycle and gain access to the auxiliary read data bus to**

transmit read data, therefore it should be clear that this teaching indicates there must have been at least one request to claim the cycle and bus from each slave device that is serving a read operation and has data to provide to the bus; the auxiliary read bus can be considered as part of the unidirectional push since it is merged with the rdDBus to feed data into rdDBusO-N ports of the master devices; see col. 4, lines 10-19 and lines 53-67 to col. 5, lines 1-42 and Fig. 10), the push bus arbiter being internal to the data processor (see col. 4, lines 1-6, PLB and its arbiter are comprised within the processor architecture), the push bus to push data from the memory resources to an input transfer memory (not clearly shown by Correale in its drawings, but it is clear that a register must be present on a processor's input port which is connected to a data bus in a processor architecture, because processors can execute instructions and fetch data much faster than memories can be accessed to provide the data, and input/output registers are needed to synchronize the data transactions between a processor and a memory resource by using a system clock) associated with the programming engines; and

a pull bus arbiter (see Fig. 10, the PLB Arbiter) to arbitrate use of a unidirectional pull bus (see Fig. 10, wrDBus) by the external memory resources in which requests for using the pull bus are sent from the memory resources (Correale also teaches the write bus can be used in place of the auxiliary read bus when the write bus is implemented as a tri-state bus, and it is clear that a tri-state bus is always unidirectional when set in a particular state; therefore it is also clear that there must have been at least one request for using the pull bus from the slave devices serving read operations since only one of them can claim the read cycle and the write bus to provide read data; see col. 5, lines 43-53; col, 4, lines 10-19 and lines 53-67 to col. 5, lines 1-42 and Fig. 10), the pull bus arbiter being internal to the data processor, the pull bus to pull data from an output transfer memory (not clearly shown by Correale, but it is clear that a register must be present on a processor's output port which is connected to a data bus in a processor architecture, because processors can execute instructions and provide read/write data much faster than memories can be accessed to provide or store the data, and input/output registers are needed to synchronize the data transactions between a processor and a memory resource by using a clock) associated with the programming engines and to transfer the data to the memory resources.

Push bus arbiter

Correale does not describe and would not have made obvious "a push bus arbiter to arbitrate requests for use of the push bus by the memory resources in which the requests for using the push bus are sent from the memory resources," as recited in amended claim 13.

Correale describes a bus system in FIG. 9 in which read and write requests from masters are arbitrated by the PLB arbiter logic, and read data, write data acknowledge, and read data acknowledge come from the slaves and are routed to particular masters (col. 4, lines 38-49). Correale then describes a system in FIG. 10 that includes an auxiliary read data bus (col. 4, line 53 to col. 5, line 17).

Correale discloses that the arbiter asserts an Auxiliary_read signal to all the slave devices during the broadcast of a read cycle, and a slave device claiming this read cycle, with Auxiliary_read asserted, may then provide read data to the auxiliary read data bus (col. 5, lines 1-7). The Examiner asserts that the slave device must send a request in order to claim the read cycle and gain access to the auxiliary read data bus. However, even if we assume that the slave device sends a signal to claim the cycle and gain access to the bus, Correale does not disclose or suggest that the signal is sent to the arbiter and arbitrated by the arbiter.

Correale states that the response to the master is the same whether a regular read cycle or an auxiliary read cycle occurred from the slave device to preserve backward compatibility (col. 5, lines 18-21). This implies that, prior to the slave device claiming the read cycle and gaining access to the auxiliary read data bus, the read request from the master has already been arbitrated (similar to the situation where the slave device gains access to a regular read data bus). It is likely that the slave device corresponding to the read address of the read request sends a signal indicating that it is ready to send the data by claiming the read cycle and gaining access to the auxiliary read data bus. Correale does not disclose or suggest that claiming a read cycle and gaining access to the auxiliary read data bus by a slave device involves sending a request by the slave device to the arbiter and having the request arbitrated by the arbiter.

Pull bus arbiter

Correale does not describe and would not have made obvious "a unidirectional pull bus to receive data from the processing agent and to transfer the data to the memory resources; a pull bus arbiter to arbitrate requests for use of the pull bus by the memory resources in which the requests for using the pull bus are sent from the memory resources," as recited in amended claim 13.

The Examiner appears to be confused about the definition of a pull bus. In claim 13, the unidirectional pull bus receives data from the processing agent and transfers the data to the memory resources. Correale discloses that a tri-state write bus can be used to transmit excess read data traffic in the same way as an auxiliary read data bus (col. 5, lines 44-49). The tri-state

write bus, when used to transmit excess read data traffic, is a "push bus", not a "pull bus" as recited in claim 13.

Even if the tri-state write bus of Correale can be considered a pull bus, Correale does not disclose or suggest that when a slave device sends a signal for claiming a read cycle and using the tri-state write bus to transmit read data, the signal is sent to the arbiter and arbitrated by the arbiter.

> **Claims 1-7, 16-21, 26-29, 33-35, 37, 40, 41, 43-45, 50, 51, 53, 54 and 57-60 are rejected under U.S.C. 103(a) as being unpatentable over Correale and Shaylor [US 6,408,325 Bl] (hereinafter "Shaylor").**
>
> **Per claims 1, 26 and 37, it is clear the Correale already teaches most of the claims as described above (the programming agent in claim 56 is considered to be equivalent to the processing agent of claim 1), and further teaches issuing a write command (the pulling of data from the processing agent to the memory resources must be a result of a write command) and loading data into an output transfer memory of the processing agent (see the rejection of claims 13, 49, 55, 56 and 61 above). Correale also discloses setting the output transfer memory to a read-only state, even though it does not explicitly recite this limitation. However, It should be clear that since processors can execute instructions and issue memory write request much faster than the actual write operations can be completed by accessing the much slower memory, the processor's output register must have its write enable control signal disabled while the data to be written to memory is ready to be transferred over the data bus, in order to avoid the data being overwritten by a new write data generated by the processor before the current transfer is complete.**
>
> **Correale does not teach executing a context. Shaylor teaches a multi-tasking and multi-threading processor that can enhance data processing efficiency (see Shaylor, col. 1, lines 46-50), the processor requires executing a context for each thread (see Shaylor, col. 1, lines 56-67). Therefore, it would have been obvious to one ordinarily skilled in the art at the time of the Applicant's invention to combine the teachings of Shaylor and Correale, in order to enhance data processing efficiency. As a result of the combination, Correale and Shaylor in combination teach "executing a context".**

Correale does not describe and would not have made obvious "using a push bus arbiter to arbitrate requests for use of a push bus by the memory resources in which the requests for using the push bus are sent from the memory resources" and "using a pull bus arbiter to arbitrate requests for use of a pull bus by the memory resources in which the requests for using the pull bus are sent from the memory resources," as recited in amended claim 1, for at least similar reasons as those applied to claim 13.

Correale also does not describe and would not have made obvious "loading data into an output transfer memory of the processing agent, setting the output transfer memory to a read-only state," as recited in claim 1.

Applicant disagrees with the Examiner's assertion that Correale discloses setting the output transfer memory to a read-only state, even though it does not explicitly recite this limitation. Correale does not even explicitly describe an output transfer memory. Even if we assume that Correale's processor has an output register, nothing in Correale suggests that the processor ever sets the output register to a read-only state. For example, it is possible that Correale's processor issues write requests at predefined write cycles such that the data in the output register are transferred out before being overwritten.

Claims 26 and 37 are patentable for at least similar reasons as those applied to claim 1.

All of the dependent claims are patentable for at least the reasons for which the claims on which they depend are patentable.

Any circumstance in which the applicant has addressed certain comments of the examiner does not mean that the applicant concedes other comments of the examiner. Any circumstance in which the applicant has made arguments for the patentability of some claims does not mean that there are not other good reasons for patentability of those claims and other claims. Any circumstance in which the applicant has amended or canceled a claim does not mean that the applicant concedes any of the examiner's positions with respect to that claim or other claims.

Please apply $1050 for the Petition for Extension of Time fee, and any other charges or credits to deposit account 06-1050, referencing attorney docket no. 10559-618001.

Respectfully submitted,

Date: 9/29/2008

Rex Huang

Rex I. Huang
Reg. No. 57,661
Attorney for Intel Corporation

Fish & Richardson P.C.
225 Franklin Street
Boston, MA 02110
Telephone: (617) 542-5070
Facsimile: (617) 542-8906